Lektion 4: Schleifen	Anwendung: Notenwarnung
Die Anwendung NOTENWARNUNG verwendet Schleifen zur Eingabe von Werten, überprüft sie auf Gültigkeit und nutzt **If** – Befehle, um passende Meldungen anzuzeigen.	**Lernziele:** • Verwendung von Zählern und Summationen. • Verwendung einer Schleife zur unbegrenzten Eingabe von Daten. • Beenden der Eingabe durch einen speziellen Wert.

Hinweis: Dieses Projekt demonstriert, wie ein Programm im Laufe seiner Entwicklung immer komplexer wird. Kontrollstrukturen und Schleifen werden ineinander geschachtelt, wobei das OS stets weiß, welches End zu welcher Struktur gehört.

Verschachtelungen

- Das ist eine Programmiertechnik, bei der Kontrollelemente oder Schleifen in andere eingebaut werden. Im Programmbeispiel geht es dabei um **If** – Befehle.
- Um Fehler zu vermeiden ist es wichtig, dass die eine Struktur vollständig in die andere eingebaut wird.
- Das Beispielprogramm zeigt eine **If** – Struktur im **Else** – Befehlsblock einer anderen **If** – Struktur. Dabei werden mehrere **End** verwendet, die der Computer dem passenden **If** zuordnet.
- Zunächst wird ein Wert für **A** eingegeben und überprüft. Ist **A<0**, so wird die Meldung „A IST NEGATIV“ angezeigt und das Programm endet. Andernfalls wird die Quadratwurzel bestimmt und in **S** abgespeichert. Jetzt kommt die zweite **If** – Bedingung (Zeilen in **Fettdruck**), in der entschieden wird, ob **A** eine Quadratzahl ist (**S** ohne Nachkommastellen) oder nicht. Anschließend wird **S** angezeigt und das Programm endet.
- Verschachtelungen werden auch benötigt, wenn in Programme Strukturen eingebaut werden, die das eigentliche Programm zwar nicht braucht, die aber den sicheren Ablauf gewährleisten wie z.B. bei der Eingabe von Werten.
- Neben dem **If** – Befehl werden hier auch gerne Schleifen verwendet.

```
prgmSQUARE
Prompt A
If A<Ø
Then
    Disp "A IST NEGATIV!"
Else
    √(A)→S
      If S=int(S)
      Then
        Disp "A IST EINE
        QUADRATZAHL."
      Else
        Disp "A IST KEINE
        QUADRATZAHL."
      End
    Disp "DIE QUADRATWURZEL
IST",S
End
```

Zusammenfassung der drei Schleifenarten:

For(*Variable,Startwert,Endwert***)**	**While <***Bedingung> ist wahr*	**Repeat** *bis <Bedingung> ist wahr*
End	**End**	**End**

For(wird verwendet beim „Zählen“ oder der Durchführung einer bestimmten Anzahl von Rechenvorgängen (Iteration).
While wird verwendet, wenn man die Möglichkeit haben will, den gesamten Befehlsblock zu überspringen.
Repeat wird verwendet, wenn man sicher ist, dass der Befehlsblock mindestens einmal ausgeführt werden soll.

Über das Schlüsselwort End

End wird bei allen Kontrollstrukturen verwendet, die über mehrere Zeilen gehen:

If … Then End	If … Then Else End	For(…) End	While … End	Repeat … End

Diese Bedeutung hat auch Einfluss auf den Aufbau des Menüs STRG (Steuerung):

7: End kommt hinter den ersten sechs Einträgen in STRG, weil es alle diese Kontrollstrukturen **End**et.

End – Befehle können vielfach in einem Programm auftauchen, aber der Computer kann jedes **End** seiner Kontrollstruktur zuordnen. Fehlt ein **End**, führt dies zu einer Fehlermeldung.

Anwendung: Programm WARNEN

Solange man in der Ausbildung ist, werden Noten erteilt, über die man auch regelmäßig unterrichtet wird. Das Beispielprogramm soll nun ermitteln, wie viele Noten eingegeben wurden und ihren Mittelwert und es soll eine Warnung ausgeben, die sich auf den Notendurchschnitt bezieht. Die Warnung soll in 3 Stufen erfolgen, deren Grenzen in Anlehnung an das Abitur festgelegt sind: unter 4 Punkten ist man „SEHR GEFAEHRDET“, unter 6 Punkten „GEFAEHRDET“ und darüber „KEINE GEFAHR“.

Es gibt zwei Methoden, wie man eine unbekannte Anzahl von Werten eingeben kann:

- Methode 1: fragt zunächst nach der Anzahl der Daten und nutzt dann eine **For(** - Schleife zur Eingabe.
- Methode 2: fragt gleich nach den Werten, verwendet aber einen besonderen Wert, der den Abbruch der Eingabe bewirkt. Bei dieser Methode wird eine **Repeat** – oder eine **While** – Schleife verwendet.

Bei der ersten Methode wird die Anzahl der Daten direkt eingegeben (ärgerlich wenn man sich verzählt hat), bei der zweiten benötigt man einen Zähler zur Bestimmung der Anzahl der Daten, denn diese Anzahl wird auf jeden Fall für die Mittelwertbildung benötigt.

Das Programm soll nun die Anzahl der Noten anzeigen, ihren Mittelwert, und es soll eine passende Mitteilung ausgeben:

Ist der Mittelwert unter 4: „SEHR GEFAEHRDET“

… 4 bis 6: „GEFAEHRDET“

… über 6: „KEINE GEFAHR“

Hinweis: Im nächsten Abschnitt werden zwei verwandte Konzepte vorgestellt: eine Zähler- und eine akkumulierende (aufsummierende) Variable. Dabei sollte man beachten, dass die Variable auf der linken Seite des Speicher-Operators zwar dieselbe ist wie auf der rechten aber bedingt durch den dahinterstehenden Prozess einen anderen Wert hat.

Zähler und Akkumulationen (Aufsummierungen)

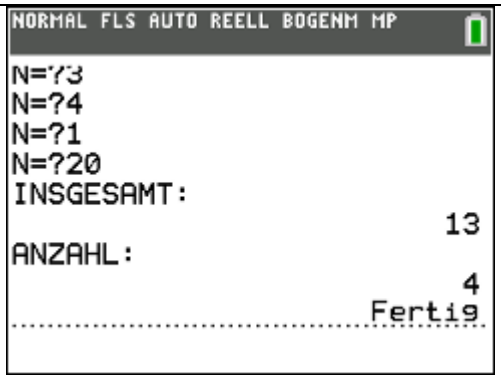
Ein Befehl wie z.B. **Z+1→Z** wird *Zähler* genannt, da er jedes Mal eine 1 zur Variablen **Z** addiert wenn er ausgeführt wird. Ein Befehl wie z.B. **A+N→A** wird *Akkumulator* genannt, da er alle Werte der Variablen **N** aufsummiert in der Variablen **A**.

Bei jeder Ausführung wird zum bisherigen Wert von **A** der neue Wert von **N** addiert und das Ergebnis wird wieder in **A** abgespeichert. Endet die Schleife, so ist in **A** die Summe aller Werte **N** gespeichert.

Das Beispielprogramm verwendet einen **Z**ähler und einen **A**kkumulator sowie ein Abbruchkriterium bei der Eingabe für die Variable **N** (**N** darf nicht 20 sein).

```
0→Z
0→N
0→A
Prompt N
While N≠20
Z+1→Z
A+N→A
Prompt N
End
Disp "INSGESAMT =",A
Disp "ANZAHL =",Z
```

Hinweise
Anfangswerte der Variablen:
N ist die Note
Z ist der Zähler
A ist die Summe
Eingabe von **N** solange **N** nicht 20
addiere 1 zum Zähler
addiere die Note zur Summe
Eingabe einer weiteren Note



Die **While** – Schleife zählt und akkumuliert die Werte für **N** bis der Wert 20 eingegeben wird. Dann wird sie verlassen und die Werte werden auf dem Display angezeigt.

Hinweis: Die beiden **Prompt** – Befehle im obigen Programm sollten noch einmal näher betrachtet werden. Der erste wird für die erste Note benötigt, der zweite für alle weiteren Noten. Außerdem befindet sich der zweite am Ende des Befehlsblockes, so dass unmittelbar darauf **N** im **While** – Befehl wieder überprüft werden kann.

Erweiterung

Bei der Eingabe könnte noch überprüft werden, ob der eingegebene Wert im Bereich von 0 bis 15 (übliches Punktesystem) liegt, wobei das Abbruchkriterium mit berücksichtigt werden muss. Bei einer Fehleingabe sollte eine Meldung auf dem Display erscheinen.

Hinweis: Die Erweiterung erfordert eine weitere Schleife für die **Input** – Befehle. Im Beispielprogramm unten wird das Programm lediglich angehalten (**Stop**), sobald eine Fehleingabe erfolgt. Das kann man besser machen!

Mögliches Programm:

```
prgmWARNEN
ClrHome
0→N
0→Z
0→A
Input "EINE NOTE EINGEBEN:",N

While N≠20
  If N<0 or N>15
  Then
    Disp "UNGUELTIG!"
    Stop
  Else
    Z+1→Z
    A+N→A
  End
  Disp "NEUE NOTE ODER ABBRUCH MIT 20"
  Input "EINGABE:",N
End
A/Z→M

ClrHome
Output(1,1,"ANZAHL DER NOTEN:")
Output(2,1,"   ZENSURENSUMME:")
Output(4,1,"      MITTELWERT:")
Output(1,19,Z)
Output(2,19,A)
Output(4,10,M)
If M<4
Then
  Output(5,9,"SEHR GEFAEHRDET")
Else
  If M<6
  Then
    Output(5,9,"GEFAEHRDET")
  Else
    Output(5,9,"KEINE GEFAHR")
  End
End
Pause
ClrHome
```

Man sollte ein Programm mit allerlei Werten testen, gerade auch mit solchen außerhalb des Gültigkeitsbereiches, und auf eventuelle Fehlermeldungen achten, die man nicht selbst programmiert hat. Fehlermeldungen sind stets ein Hinweis darauf, dass man entweder einen syntaktischen oder einen logischen Fehler gemacht hat, also nicht alle Möglichkeiten abgesichert hat. Das Programm enthält so einen Fehler. Gibt man als ersten Wert 20 ein, so wird der Befehlsblock der **While** – Schleife sofort übersprungen. In der Zeile A/Z→M würde dann 0 durch 0 dividiert, eine nicht definierte Division. Man kann diesen Fehler beheben, indem man dafür sorgt, dass wenigstens eine Note eingegeben wurde, etwa durch die folgenden Zeilen

```
If Z>0
Then
 A/Z→M
Else
 Disp “KEINE NOTE EINGEGEBEN!”
 Stop
End
```

Der Befehl **Stop** beendet ein Programm augenblicklich. Man kann ihn überall in ein Programm einfügen, so dass das Programm in seiner weiteren Ausführung beendet wird und **Done** auf dem HOME – Bildschirm erscheint.